MySQL HA

Lösungen für Front- und Backend

Matthias Klein

About InnoGames

- Betreibt und entwickelt Browser- und Mobile Games seit 2007 für über 200 Millionen Spieler
- Mehr als 400 Mitarbeiter aus über 30 Nationen sind an den Standorten Hamburg und Düsseldorf tätig

About Me

- Seit 2009 als Administrator in der Gaming Industrie tätig
- Zuständig für das Payment System
- Ansprechpartner für MySQL, Mail, ELK

1. Grundsätzliches
2. Backend
3. Frontend
4. Migrationspfade

Was ist eigentlich...

- HA: High Availability – Im Idealfall ist es uns egal, was ausfällt
- Frontend: Die Schnittstelle unserer Datenbank(en) zu unserer App
- Backend: Unsere Datenbanken

Sysadmins Helferlein

- Percona toolkit: pt-online-schema-change für DDL
- Percona Xtrabackup: konsistente Backups ohne Downtime

Flavours of MySQL

PERCONA
Server for MySQL

MySQL != InnoDB

- MySQL unterstützt unterschiedliche Engines, die jeweils ihre eigenen Vor- und Nachteile haben
- InnoDB ist eine sehr gute „All Purpose“ Engine

MySQL Engines - MyISAM

- Text Indizes
- Schnelle SELECTs
- Nicht transaktionssicher
- neigt zur Selbstzerstörung

MySQL Engines - Blackhole

- Null device
- Sehr nützlich bei der klassischen Replikation

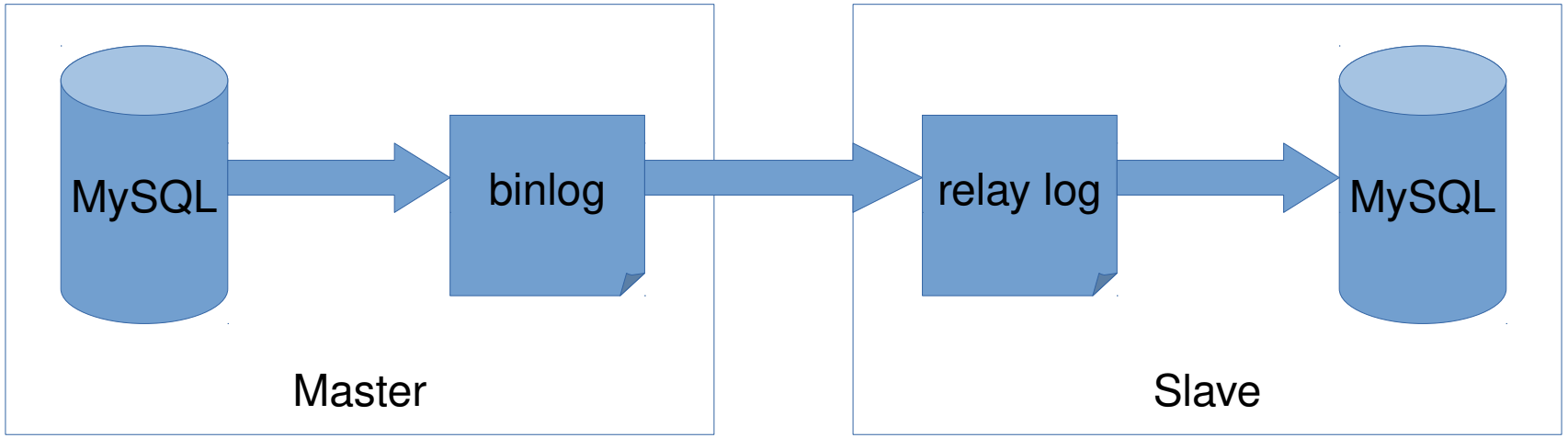
MySQL Engines - Archive

- Automatische Kompression
- Unterstützt ausschließlich INSERT / REPLACE / SELECT
- Ein SELECT verursacht immer einen full table scan

Klassische Replikation

- Verfügbar seit 2001
- Asynchrones Verfahren
- Wird häufig optimiert und erhält neue Features

Klassische Replikation - Funktionsweise



Klassische Replikation – Features

- Replikationsketten und -ringe sind möglich (Master – Master, Intermediate Master)
- Unterschiedliche Filterregeln auf Master und Slave
- Unterschiedliche Engines auf Master und Slave
- Unterschiedliche Datenbank- / Tabellennamen auf Master und Slave

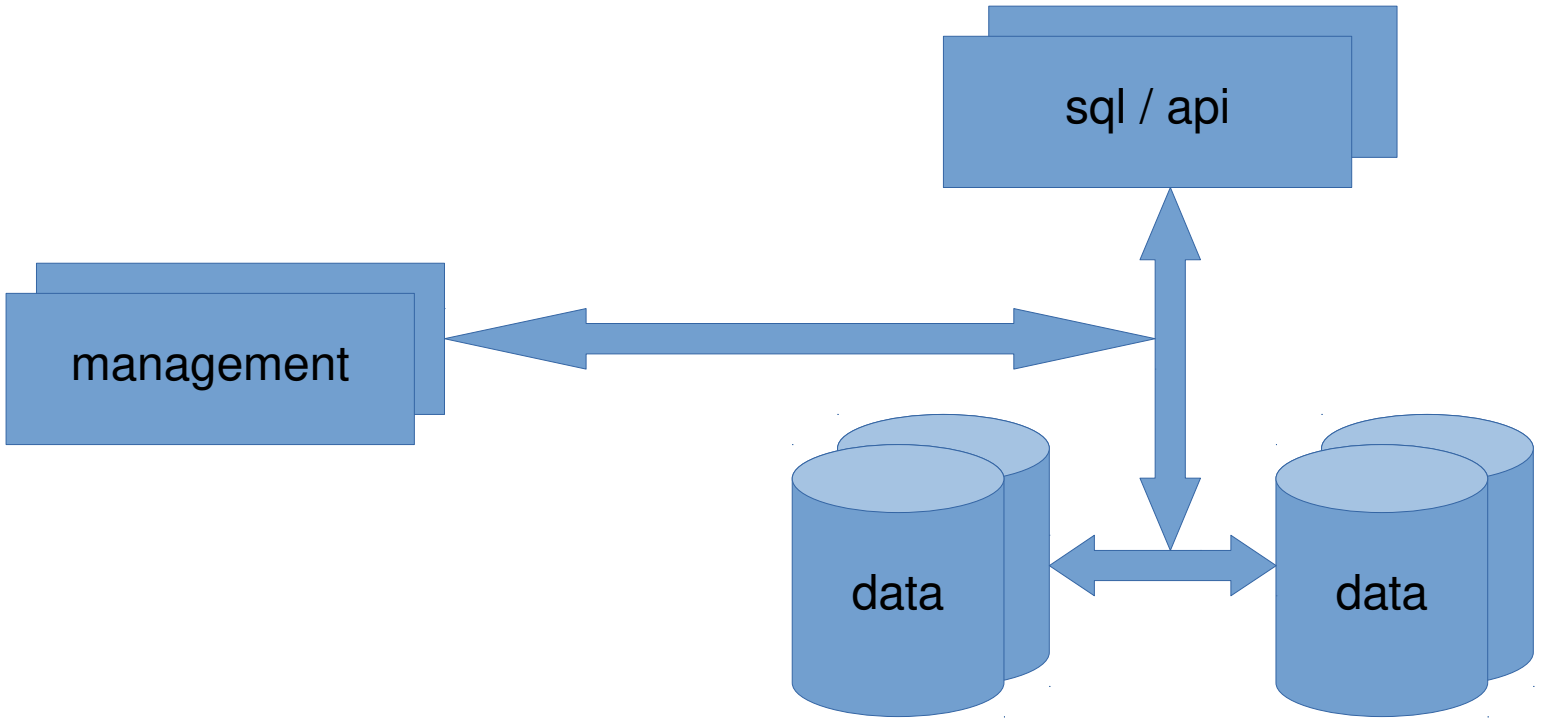
Klassische Replikation – Klassische Probleme

- Slave Lag → slave_parallel_workers
- binlog_format STATEMENT / MIXED / ROW
- Binlog Positionen bei Slave Promotion → GTID / Master - Master
- Aufsetzen eines Slaves bei laufendem Produktivbetrieb → Percona Xtrabackup
- Schreiben auf mehreren Hosts

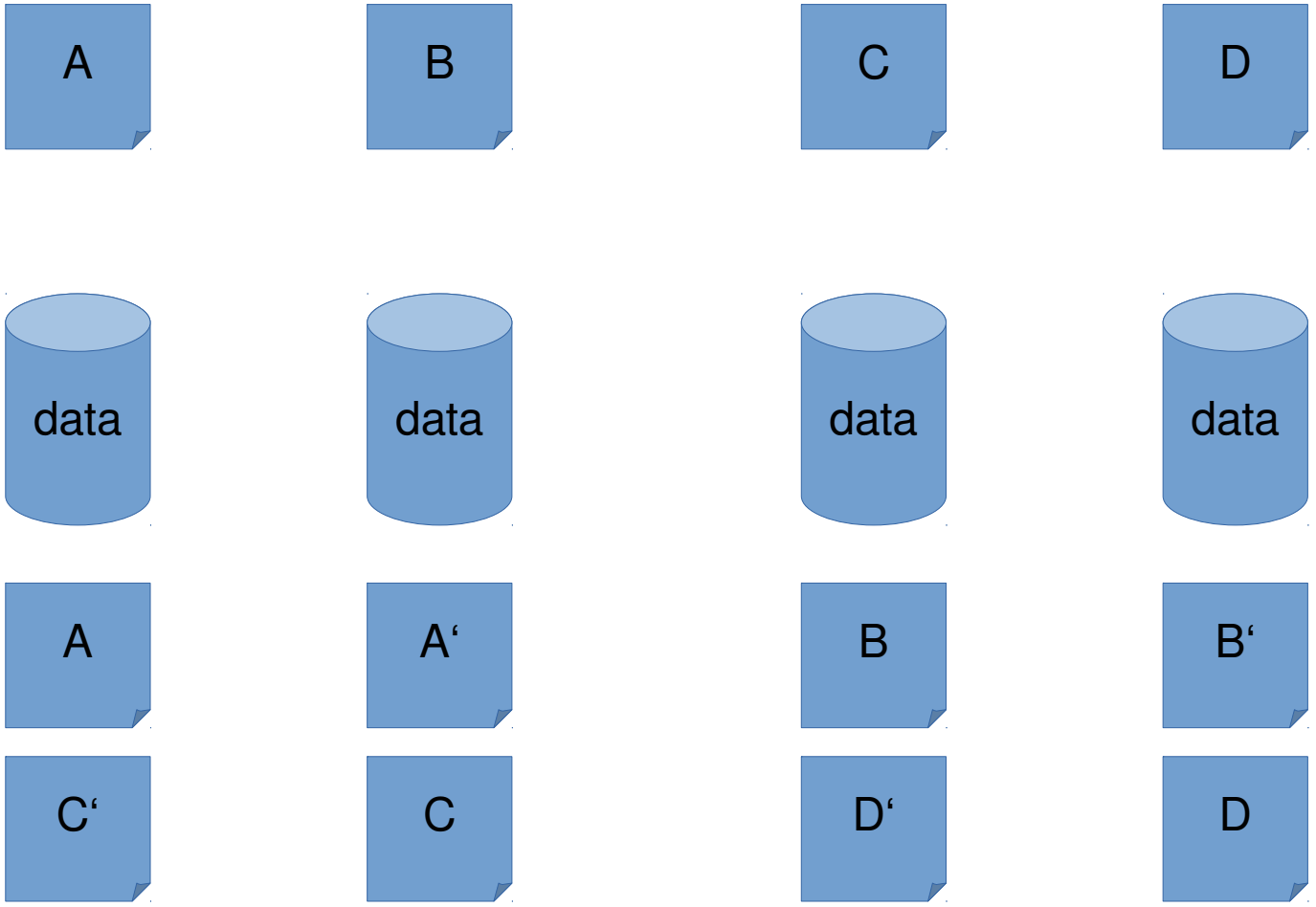
ndb cluster

- Verfügbar seit 2001
- Synchrones Verfahren
- Multi Master, Auto Sharding, (fast) ohne SpoF
- In-Memory und/oder On-Disk

ndb cluster - Funktionsweise



ndb cluster - Funktionsweise



ndb cluster - Nachteile

- Durch die Architektur sind bestimmte Operationen extrem teuer (z.B. join / group by / limit), wird aber stetig optimiert.
- Verursacht sehr hohe Netzwerkauslastung, ein dediziertes >= 1Gb/s Netzwerk für die data nodes ist ratsam

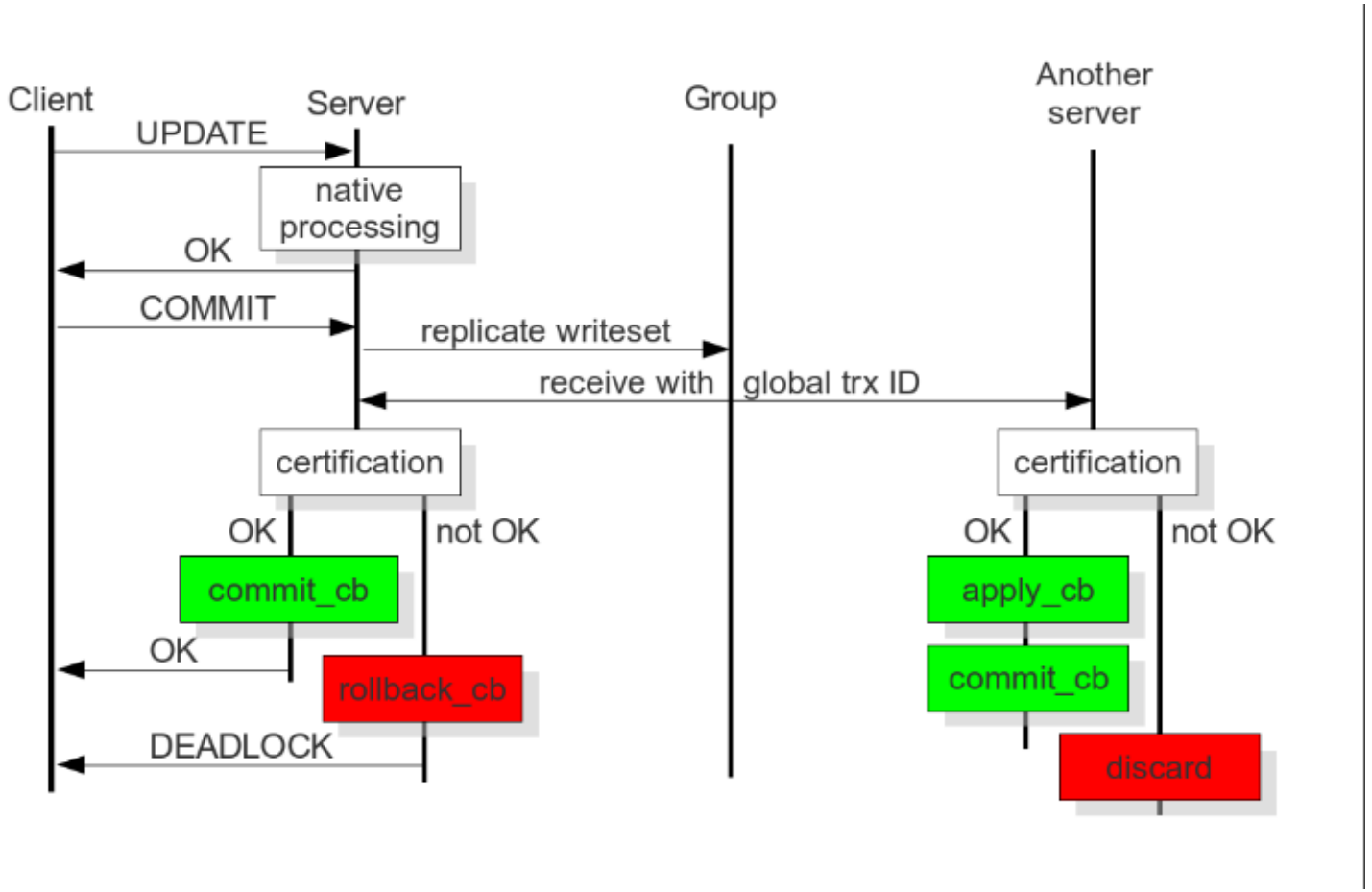
Galera Cluster / MySQL Group Replication

- Virtuell Synchrones Verfahren
- Multi Master ohne Sharding
- Automatische Synchronisation neuer Nodes mit dem Cluster
- Es wird eine ungerade Anzahl nodes benötigt (Split Brain)

Galera Cluster / MySQL Group Replication

- Bootstrap für den ersten Node nötig (Wichtig, wenn alles kaputt ist)
- „Single Primary Mode“ empfehlenswert
- Galera ist besser bei WAN-Replikation
- Galera hat den Vorteil des früheren Starts, MySQL kann seine Lösung besser ins System integrieren

Galera Cluster / MySQL Group Replication - Funktionsweise



Galera Cluster / MySQL Group Replication - Unterschiede

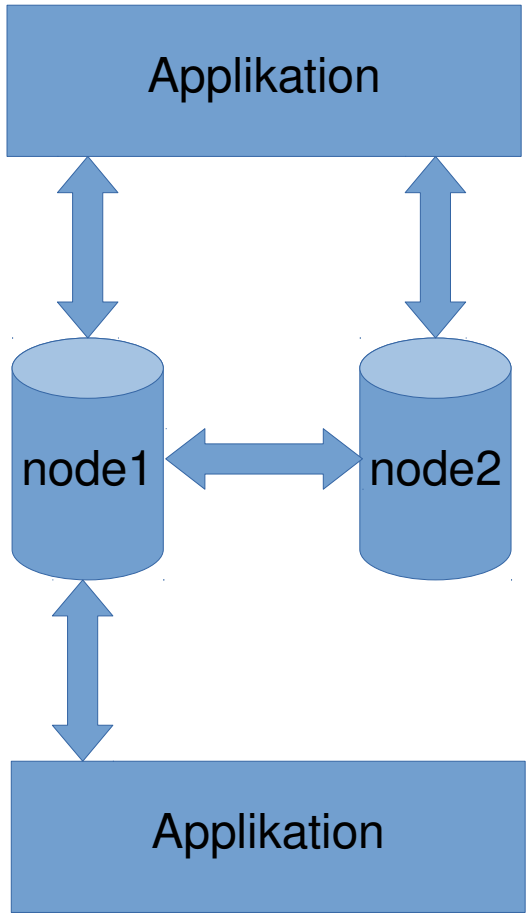
- Bei Konflikten liefert Galera einen „Deadlock“ zurück
- Gruppenkommunikation bei Galera über Totem Single Ring, MySQL über Paxos → Galera wird mit mehr nodes langsamer
- Arbiter gibt es bei MySQL (noch) nicht

Was soll ich wählen?

Kommt drauf an:

- Ich muß sharden: ndb cluster
- Ich habe große Writesets (>128k/row, >1GB total): klassische Replikation
- Ich möchte ein robustes System: Cluster

Warum Frontend HA?



Frontend HA - Möglichkeiten

- „Virtuelle“ IPs mit Manager (mmm, mha, orchestrator)
- Proxy / Loadbalancer (ProxySQL, haproxy, testtool)

Virtuelle IPs - Funktionsweise

- Zur Laufzeit wird die IP, auf die die Applikation sich verbindet, auf einen Datenbankhost gebunden
- Im Fehlerfall wird diese IP auf einen anderen Host gebunden
- Problem: Trennen bestehender Verbindungen (mmm)
- Problem: Es darf nur einen Manager geben → StandBy möglich

MySQL-Multi-Master-Monitor (mmm)

Master High Availability Manager (mha)

- Wird jetzt wieder weiterentwickelt
- Kann jetzt auch GTID und IPv6
- Bringt eigene Tools zum Monitoring und Failover mit
- Output ist mit Nagios nutzbar
- Klare Empfehlung für eine einfache Lösung ohne SchnickSchnack

Orchestrator

- Unter ständiger Entwicklung
- Nettes Frontend zum Monitoring und Management
- Scripts / Tools zum Failover müssen selbst erstellt werden
- Auto Discovery von hosts

ProxySQL

- Hat Failover nur als Nebeneffekt, muß mit anderen Tools vervollständigt werden (z.B. keepalived)
- Query Caching
- Query Routing
- Query Firewall

haproxy

- Monitoring Scripte müssen selbst erstellt werden (Beispiele vorhanden), werden meist über xinetd getriggert
- Eigener Failover wird über keepalived realisiert
- Ist für Galera / Group Replication geeignet
- Probleme mit Permissions in der Datenbank möglich (TCP-Forward)

Loadbalancing mit pf (Eigenentwicklung)

- Loadbalancer mit Failover (BGP) sind schon vorhanden
- Tool zur Steuerung von pf wird schon lange für Webserver eingesetzt
- Scripte für MySQL (xinetd) müssen selbst erstellt werden

Was soll ich wählen?

Kommt drauf an:

- Cluster: haproxy oder Eigenlösung
- Klassische Replikation: mha oder orchestrator
- Ich brauche ProxySQL features: ProxySQL + keepalived

Migrationspfade

- Im Frontend relativ einfach, da
  - Tools zum IP Management besitzen einen „observer“ Modus
  - Bei Proxy / Loadbalancer kann die Applikation umgestellt werden
- Im Backend nur dann ohne Downtime, wenn das binlog schon an ist

Binlog Positions zu GTID

- In MySQL 5.7 online möglich
- In MySQL 5.6 (Percona) über Zwischenschritte online möglich:
  - gtid_mode=ON; gitd_deployment_step=ON; auf den slaves
  - Einen Slave mit gtid_mode=ON; gitd_deployment_step=OFF; zum Master promoten
  - Restliche Server auf gtid_mode=ON; gitd_deployment_step=OFF; stellen

Asynchron zu Synchron (und zurück)

- Die Replikationsarten lassen sich mischen
- Das Zielsystem wird komplett (mit den Daten) aufgesetzt und läuft als Slave mit (Master – Master für mutige)
- Switch erfolgt über das Frontend

NOCH FRAGEN?

VIELEN DANK FÜR EURE AUFMERKSAMKEIT!

INNOGAMES @GITHUB:

WWW.GITHUB.COM/INNOGAMES

INNOGAMES IS HIRING:

WWW.INNOGAMES.COM

 @innogames  /innogames  /innogames

 matthias.klein@innogames.com